Course Notes

Matthew Finlayson
CS183 — Fundamentals of Machine Learning (FML)

April 20, 2020

Contents

1 Introduction

Premise: Machine Learning follows a deep and rigorous theory that can be harnessed to create magic in AI and the data sciences.

1.1 Mathematical model for statistical learning

- Example x: are drawn i.i.d form some unknown prob dist D

- There is a deterministic function $f : X \to Y$ s.t. $\forall(x_i, y_i) \in X \times Y$, $y_i = f(x_i)$

1.2 Measuring success of a hypotheses h

- Given h the loss can be measured as:

$$L_D(h) := P_{x \sim D}[h(x) \neq f(x)]$$

  The loss of h on D.

1.3 How do we find h?

1.3.1 Empirical Risk Minimization

Design h such that it minimizes the empirical loss, where the empirical loss is the number of $(x_i, y_i) \in S$ s.t. $h(x_i) \neq y_i$.

$$L_S(h) := \frac{|\{x_i \in S_x : h(x_i) \neq f(x_i)\}|}{m}$$

Find $h_s \in \operatorname{argmin} L_S(h)$

1.3.2 Why could ERM be a bad idea?

$$h(x) = \begin{cases} y_i & \forall x_i \in S_x : x = x_i \\ 0 & \text{o.w.} \end{cases}$$

Empirical loss of h_S: $L_S(h) = 0$

True loss of h_S: $L_D(h) = \frac{1}{2}$. This is just flipping a coin. This is refered to as **overfitting**.

1.4 Inductive bias

Restrict the search space of possible classifiers to some predefined class $\mathcal{H}$.

$$ERM_{\mathcal{H}}(S) := \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$$

This could be the set of lines through a point, set of quadratic equations, neural networks, or anything.

Theorem 1 (PAC learnability)**.** Let $\mathcal{H}$ be finite and suppose that $\mathcal{H}$ is good in that $\exists h^* \in \mathcal{H}$ s.t. $L_D(h^*) = 0$ (realizability assumption). Given $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ where $x_i \sim D$ i.i.d., let $h_S \in \operatorname{argmin}_{h \in \mathcal{H}} L_S(h)$. Then $\forall \epsilon, \delta \in (0, 1)$ with probability $1 - \delta$ over S: $L_D(h_S) \leq \epsilon$ when $m \geq \log(|\mathcal{H}|/\delta)/\epsilon$.

Proof. Suppose $h_S \in \operatorname{argmin}_{h \in \mathcal{H}} L_S(h)$ but h_S has error $\geq \epsilon$. A hypothesis is bad if its ture error $L_D(h) \geq \epsilon$; $\mathcal{H}_B$ is the set of all bad hypotheses in $\mathcal{H}$. A sample S is **misleading** if $\exists$ bad hypothesis s.t. $\forall x \in S$, $h(x) = h^*(x)$. For a given bad hypothesis, what is the probability that h classifies a point correctly is $< 1 - \epsilon \leq e^{-\epsilon}$. The likelihood of bad classifier correctly calssifying m points is $< (1 - \epsilon)^m < e^{-\epsilon m}$. The probability of S being misled is $|\mathcal{H}_B| e^{-\epsilon m} \leq |\mathcal{H}| e^{-\epsilon m} = \delta$. □

2 Learnability

2.1 Recap

Mathematical model for statistical learning

- Sample $S = \{(x_1, y_1), \ldots (x_m, y_m)\}$ of m examples where $x_i \sim$ i.i.d. from **unknown** dist. D such that $x \in X$, $y \in Y$.
- Labeling funciton $f : X \to Y . \forall y \in Y \; y_i = f(x_i)$.

- Goal: Design $h : X \to Y$ that is $L_D(h) \leq \epsilon$ for any $\epsilon \in (0, 1)$ with probability $1 - \delta$.
- Measuring performance of classifier h:

$$L_D(h) := \Pr_{x \sim D}[h(x) = y]$$

- Empirical loss:

$$L_S(h) = \Pr_{x \sim Su.a.r.}[h(x) \neq y]$$

- Minimizing empirical loss can lead to overfitting.
- Inductive bias: Minimize empirical loss over some hypothesis class $\mathcal{H}$. Restricting search space to $\mathcal{H}$ can mitigate overfitting.

$$ERM_{\mathcal{H}}(S) := \min_{h \in \mathcal{H}} L_S(h)$$

- Realizability assumption: $\exists h^* \in \mathcal{H}$ s.t. $L_D(h^*) = 0$.

2.2 PAC learnability

Definition 1 (PAC learnable)**.** A hypothesis class $\mathcal{H}$ is PAC learnable if there exists function $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and learning algorithm with the following property: $\forall \epsilon, \delta \in (0, 1)$, every dist. D, every labeling function $f : X \to \{0, 1\}$, if the realizability assuption holds w.r.t $\mathcal{H}$, f, D, then running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ examples drawn i.i.d. from D the algorithm returns $h : X \to Y$ s.t. $L_D(h) \leq \epsilon$ w.p. $1 - \delta$.

In other words

$$\Pr[L_{D,f}(h) < \epsilon] \geq 1 - \delta \Leftrightarrow |S| \geq m_{\mathcal{H}}(\epsilon, \delta)$$

Probably (δ, confidence) Approximately (ϵ, approximation or precision) Correct ($L_D(h) = 0$).

(PAC learnability theorem) Given realizability, every finite hypothesis class is PAC learnable, and the sample complexity is at most $m_{\mathcal{H}}(\epsilon, \delta) = \log(|\mathcal{H}|/\delta)/\epsilon$.

2.3 Agnositc PAC learnability

We want to remove the realizability assumption. There are classes that will not be perfectly learnable. We therefore set a new goal.

New goal: Find the h s.t. $L_D(h) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$.

What if f is not deterministic? Instead there is a probability distribution, each x has a some probability that $y = 1$.

Let D be a joint probability distribution over $X \times Y$.

$$L_D(h) = \Pr_{(x,y) \sim^{i.i.d.} D}[h(x) \neq y]$$

$$L_S(h) = \Pr_{(x,y) \sim^{\text{u.a.r.}} S}[h(x) \neq y]$$

Agnostic - the labels are from a distribution, not from a "higher power" f.

Example 1. Not classifying, but predicting (height vs. shoe size). This is a **regression problem** (as opposed to a **classification problem**.) We want to have a general way of talking about both.

In classification, $L_D(h) = \Pr[h(x) \neq y]$.

$$l_{0-1}(h, (x, y)) = \begin{cases} 0 & h(x) = y \\ 1 & h(x) \neq y \end{cases}$$

In regresssion: $L_D(h) = \mathbb{E}[(h(x) - y)^2]$

$$l_{sq}(h, (x, y)) = (h(x) - y)^2$$

Generalized loss functions:

$$L_D(h) = \mathbb{E}_{z \sim D}[l(h, z)]$$

$$L_S(h) = (1/m) \sum_{i=0}^{m} l(h, z_i)$$

Examples: $z \in X \times Y$

Definition 2 (Agnostic PAC learnability)**.** A hypothesis class $\mathcal{H}$ is agnostic-PAC-learnable w.r.t. a set $Z = X \times Y$ and loss function $l : \mathcal{H} \times Z \to \mathbb{R}$ if there exists some $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and learning alg with the following property: For all $\epsilon, \delta \in (0, 1)$, every dist. D over Z, when running alg on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ examples drawn i.i.d. from D the alg returns h s.t.:

$$L_D(h) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$$

with probability $1 - \delta$.

2.4 Learnability via uniform convergence

Definition 3 (ϵ-representative)**.** A sample S of examples drawn i.i.d. from D is **ϵ-representative** if for all $h \in \mathcal{H}$, $|L_S(h) - L_D(h)| \leq \epsilon$.

Definition 4 (Uniform convergence)**.** $\mathcal{H}$ has **uniform convergence** property if for all $\epsilon, \delta \in (0, 1)$ and every probability dist. D over Z, if S is $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ examples drawn i.i.d. from D then S is ϵ-representative.

Theorem 2 (Agnostic PAC learnability)**.** If $\mathcal{H}$ has uniform convergence property then $\mathcal{H}$ is agnostically PAC learnable with sample complexity $m_{\mathcal{H}}(\epsilon/2, \delta)$. Furthermore, $ERM_{\mathcal{H}}$ paradigm produces the hypothesis.

Proof. Let $h_S \in \text{argmin}_{h \in \mathcal{H}} L_S(h)$.

$$L_D(h_S) \leq L_S(h_S) + \epsilon/2 \leq \min_{h \in \mathcal{H}} L_S(h) + \epsilon/2 \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$$

□

Lemma 2.1. Every finite hypothesis class has the uniform convergence property.

Proof. Fix some ϵ, δ. We need to prove

$$\Pr[\{S : \forall h \in \mathcal{H} \text{s.t.} |L_S(h) - L_D(h)| \leq \epsilon\}] \geq 1 - \delta$$

or

$$\Pr[\{S : \exists h \in \mathcal{H} \text{s.t.} |L_S(h) - L_D(h)| > \epsilon\}] < 1 - \delta$$

This probability is bounded by

$$\sum_{h \in \mathcal{H}} \Pr[S : |L_S(h) - L_D(h) > \epsilon]$$

Reminder:

- $$L_D(h) = \mathbb{E}_{z \sim D}[l(h, x)]$$
- $$L_S(h) = (1/m) \sum_{i=0}^{m} l(h, z_i)$$

- Any given h is fixed, z is drawn i.i.d., thus $l(h, z)$ also drawn i.i.d..
- Every z_i is drawn i.i.d. therefore $l(h, z_i)$ has mean $\mathbb{E}_{z \in D}[l(h, z)] = L_D(h)$ and linearity of expectation says $\mathbb{E}[L_S(h)] = L_D(h)$.

By the Hoeffding bound where $\theta_i = l(h, z_i)$, $L_S(h) = (1/m)\sum_{i=1}^{m}\theta_i$. $L_D(h) = \mu$.

$$\sum_{h\in\mathcal{H}} \Pr[S : |L_S(h) - L_D(h)| > \epsilon] \leq |\mathcal{H}|2e^{-2m\epsilon^2}$$

$$m_{\mathcal{H}} = \log(2|\mathcal{H}|/\delta)/(2\epsilon^2)$$

□

Hoeffding bound: Let $\theta_1, \ldots, \theta_m$ be a sequence of r.v. drawn i.i.d. and assume for all $\mathbb{E}[\theta_i] = \mu$ and $\Pr[a \leq \theta_i \leq b] = 1$ then for all ϵ: $\Pr[|1/m\sum_{i=1}^{m}\theta_i - \mu| > \epsilon] \leq 2e^{-2m\epsilon^2} = \delta$.

2.5 Section

2.5.1 Perceptron

The perceptron algorithm is a learning algorithm tha timplememts the ERM rule fo binary classification tasks. It ouputs a predictor of the form $\mathbf{w} = (w_0, \ldots, w_n)$.

3 VC-dimensionality

(the peak of theoretical obscurity)

3.1 PAC learnability recap

- Mathematical model for statistical learning:
  - Given $S = \{(x_1, y_1), \ldots, (x_,y_m)\}$
  - Assuming $\exists$ distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.

– And each example $s \in S$ is drawn i.i.d. from $\mathcal{D}$.

– Goal: design alg that produces hypothesis $h : \mathcal{X} \to \mathcal{Y}$ s.t.

$$L_{\mathcal{D}}(h) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

with probability $1 - \delta$.

- True error: $L_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$
- Empirical error: $L_S(h) = \Pr_{(x,y)\overset{uar}{\sim}S}[h(x) \neq y]$
- Proved: Every finite hypothesis class H is PAC learnable.

3.2 Infinite-size classes can be learnable

Example 2. Let H be theclass of threshold functionsin $\mathbb{R}$.

$$H = \{h_a : a \in \mathbb{R}\}$$

$$h_a(x) = \begin{cases} 1 & x < a \\ 0 & x \geq a \end{cases}$$

This is a linear classifier in 1 dimension. $|H| = \infty$, but we will show that H is PAC learnable.

Lemma 2.2. Let H be the class of threshold funcitns. Then H is PAC learnable using the ERM rule with sample complexity $m_H \leq \log(2/\delta)/\epsilon$.

Proof. Let a^* be s.t. $L_D(h_{a^*}) = 0$. There exist $a_0 \leq a^* \leq a_1$ s.t. the probability masses from a_0 to a^* and also from a^* to a_1 are ϵ. Given S, b_0 is the largest example with label 1, b_1 is the smallest example with label 0. Choose b_s such that $b_0 \leq b_s \leq b_1$. Therefore $L_D(h_{b_s}) = 0$, therefore it is an ERM. Letting

$$\Pr[L_S(h_{b_s}) > \epsilon] \leq \Pr[b_0 < a_0 \text{ or } b_1 > a_1]$$

or

$$\leq \Pr[b_0 < a_0] + \Pr[b_1 > a_1]$$

We have that

$$\Pr[b_0 < a_0] = \Pr[\forall x \in S_x, x \notin (a_0, a^*)] = (1-\epsilon)^m \leq e^{\epsilon m}$$

(Use symmetry for other side.) We can thus find an upper bound on the sample size complexity. □

This is tedious specific to one hypothesis class. How can we generalize?

3.3 The VC dimension

Named after Vaphic and Chervonekis who came up with the idea in 1970 (50 years ago!)

Theorem 3 (No Free Lunch)**.** If we don't restrict the hypothesis class, then we cannot learn. In other words without a restriction on the hypothesis class H, for any algorithm, an adversary can construct a solution s.t. the algorithm does not have an error less than ϵ with probability greater than $1 - \delta$.

Definition 5 (Restriction)**.** Let H be a class of functions from $\mathcal{X}$ to $\{0, 1\}$ and $C \subseteq \mathcal{X}$ where $|C| = m$. The restriction of H on C, denoted H_C is the set of all functions we can derive from H from $C \to \{0, 1\}$.

$$H_C := \{(h(C_1), \ldots, h(C_m)) : h \in H\}$$

Thus an element of H_C can be written as a string in $\{0, 1\}^m$.

Definition 6 (Shattering)**.** A hypothesis class H shatters C if H_C is the set of all functions $C \to \{0,1\}$, or in other words $|H| = 2^{|C|}$.

If no such C exists, $VCdim(H) = \infty$

Example 3 (Shattering)**.** Let $C = \{C_1\}$, H is the class of threshold functions. Does H shatter C? $H_C = \{0,1\}$, therefore yes.

Now let $C = \{C_1, C_2\}$ where $C_1 < C_2$. Does H still shatter C? No because $01 \notin H_C$.

Definition 7 (VC dimension)**.** The VC dimension of a hypothesis class H denoted $VCdim(H)$ is the size of the maximal size of the set C that can be shattered by H.

Corollary 3.1. Let H be a hypothesis class form $\mathcal{X}$ to $\{0,1\}$. Let m be a training set size. Assume there is $C \subset \mathcal{X}$ of size $2m$ that is shatteredby H. Then forany learnying algorithm A there exists a distribution D over $\mathcal{X} \times \{0,1\}$ and predicotr $h \in H$ s.t. $L_D(h) = 0$ but with probability greater than $1/7$ over chnoice of $S \sim D$ we have $L_D(A(S)) \geq 1/8$ despite realizability.

Theorem 4 (Finite VC dimensional requirement for PAC learnability)**.** Let H be a class with inifinte VC dimension. Then H is not PAC learnable.

Proof. H has infinite VC dimension. Therefore for any training size m there exists some set C of size $2m$ shattered by H. By corollary, not PAC learnable. □

It turns out that the converse of our theorem is true.

Example 4. To prove the VC dimension of a hypothesis class H is d:

1. Prove there is a set of size d that is shattered by H.

2. Prove all sets of size $d+1$ are not shattered by H.

Prove VC dimensionality of

- $H_{a,b}$ is the set of intervals

$$\{h_{ab} : a, b \in \mathbb{R}, a \leq b\}$$

$$h_{ab} = \begin{cases} 1 & x \in (a, b) \\ 0 & \text{o.w.} \end{cases}$$

  Answer: 2

- Rectangles H_{a_1,a_2,b_1,b_2}. Answer: 4

Theorem 5 (PAC learnability from finite VC dimensionality)**.** If H has finite VC dimension then H has the uniform convergence property.

Proof idea. If $VCdim(H) = d$ then $|H_C| = O(|C|^d)$ (Sauer's Lemma.) If $|H_C| = O(|C|^d)$ then uniform convergence holds. Full proof can be found in chapter 6 (look for proofs with astrisk.) □

3.4 Section

Hints for proving VC dimensionality.

1. Find a 'convex hall' such that there exists a point within certain bounds.
2. Generalize this

4 Linear predictors, half spaces, regression

4.1 Recap

Given S of lenght m, find $h : X \to Y$ where $s \in S$ is drawn i.i.d from unk distribution D such that $L_D(h) \leq \min_{h \in H} L_D(h) + \epsilon$ with probability greater than $1 - \delta$.

See L_D definition for classification from lecture 2. We also have L_S, the empirical loss (or risk). We try to minimize L_D by minimizing L_S using an ERM.

$$ERM(S) = \underset{h \in H}{\operatorname{argmin}} L_S(h)$$

H is PAC learnable iff VC dimension of H is finite. If VC dimension of H is finite, H is PAC learnable via ERM.

What is the connectoin between ERM and PAC learnability.

4.1.1 Sample complexity

$X \in \mathbb{R}^a$, $Y \in \{0, 1\}$. If H has VC dimension d then H is agnostic PAC learnablie with

$$C_1 \left(\frac{d + \log(1/\delta)}{\epsilon^2} \right) \leq m_H(\epsilon, \delta) \leq C_2 \left(\frac{d + \log(1/\delta)}{\epsilon^2} \right)$$

and H is PAC learnable (assuming realizability) with

$$C_1 \left(\frac{d + \log(1/\delta)}{\epsilon} \right) \leq m_H(\epsilon, \delta) \leq C_2 \left(\frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon} \right)$$

We want to minimize sample complexity.

4.2 Linear predictors

Prediction rules based on linear functions. Easy to do ERM on these.

The class of affine functions:

$$L_d = \{h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where

$$h_{w,b}(x) = \langle w, x \rangle + b = w^\top x + b = \sum_{i \in [d]} w_i x_i + b$$

The class of homogeneous linear functions:

$$L'_d = \{h_w(x) : w \in \mathbb{R}^d\}$$

where $h_w(x) = \langle w, x \rangle = w^\top x$

Any affine function can be rewritten as a homogeneous fn by appending 1 to x and b to w.

Hypothesis classes using linear functions:

$$\varphi \circ L$$

where $\varphi : \mathbb{R} \to Y$ in class funciton

$$\varphi(z) = \text{sign}(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

4.3 Halfspaces

Example 5. $X \in \mathbb{R}^2$, $Y \in \{-1, 1\}$.

Definition 8 (Halfspace)**.** Halfspaces are the regions separated by a hyperplane (plane of dimension $d-1$.)

$$HS_d = sign \circ L_d = \{x \to \text{sign}(h_{wb}(x)) : h_{wb} \in L_d\}$$

What is the sample complexity?

Theorem 6. The class of homogeneous halfspaces from $X \in \mathbb{R}^d$ to $Y \in \{0,1\}$ has VC dimensoin d. Therefore the sample complexity is

$$m(\epsilon, \delta) \leq C(\frac{d + \log(1/\delta)}{\epsilon^2}$$

Proof.

1. We show that there is a set C of size d shattered by H. consider basis vectors $e_1, \ldots, e_d$. I assert that this set is shattered by HS_d. For all labeling $y_1, \ldots, y_d$, consider $w = (y_1, \ldots, y_d)$. $w^\top e_i = y_i$ for all i. Therefore HS_d shatters C of size d.

2. Consider $x_1, \ldots, x_{d+1}$. Exists $a_1, \ldots, a_{d+1}$ s.t. $\sum_{i \in [d+1]} x_i a_i = 0$. where $a_i \neq 0$. Define $I = \{i : a_i > 0\}$, $J = \{j : a_j < 0\}$. Assume that $x_1, \ldots, x_{d+1}$ are shattered by HS_d. So there is a w s.t. $w^\top x \geq 0$ foor all $i \in I$ and $w^\top x \leq 0$ for all $j \in J$. Lets assume both I and J are nonempty.

$$\sum_{i \in I} a_i x_i = \sum_{j \in J} |a_j| x_j$$

$$0 < \sum_{i \in I} a_i \langle w, x_i \rangle$$

$$\ldots$$

   Show both I and J cannot be nonempty. Contradiction.

□

Now all we need to do is find a poly-time algorithm for ERM on halfspaces.

Goal: Design algorithm for solving $ERM(S)$ when H is the class of halfspaces. Today: Realizable case. Linear programming. Perceptron.

4.4 Linear programming

LP is a framework for solving problems that can be formulated as find $\max u^\top x$ such that $Ax \geq v$ where $w \in \mathbb{R}^d$, A is a $m \times d$ matrix, v is in $\mathbb{R}^m$, in polynomial time.

4.4.1 ERM using LPs in the realizable case

Given S of size m. We are looking for $w \in \mathbb{R}^d$ s.t. $\text{sign}(w^\top x_i) = y_i$ for all $i \in [m]$.

Equivalently, we want $y_i \langle w, x_i \rangle > 0$ for all $i \in [m]$.

Finding w using LPs: Let w^* be the vector that respects $y_i \langle w^*, x_i \rangle > 0$ for all i. By realizability, we know that this exists. Define $\gamma = \min_{i \in [m]} y_i \langle w^*, x_i \rangle$. Define $\bar{w} = (w^*/\gamma)$. We have that

$$y_i \langle \bar{w}, x_i \rangle = (1/\gamma) y_i \langle w^*, x_i \rangle \geq 1$$

There exists some $\bar{w}$ s.t. $y_i \langle \bar{w}, x_i \rangle \geq 1$ for all $i \in [m]$.

$A_{ij} = y_i x_{ij}$, $v = \{1\}^m$. Use dummy objective $u = \{0\}^d$. Solve LP $\max u^\top w$ s.t. $Aw \geq v$ to get separating hyperplane.

4.5 Regression

$$H_{reg} = L_d = \{x \to \langle w, x \rangle + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Loss function:

$$l(h(x, y)) = (x - y)^2$$

ERM for regression:

$$L_S(h) = (1/m) \sum_{i \in [m]} (h(x) - y)^2$$

Find $\text{argmin}_{h \in H} L_S(h)$.

Alternatively $l(h(x, y)) = |h(x) - y|$ which can be solved with LP.

For single dimension:

$$\min_{w \in \mathbb{R}} (1/m) \sum_{i \in [m]} (y_i - w x_i)^2$$

$$g(w) = (1/m) \sum_{i \in [m]} (y_i - w x_i)^2$$

Goal find $\text{argmin}_{w \in \mathbb{R}} g(w)$. In pieces, find $\text{argmin}_w (y_i - w x_i)^2$.

$$\partial (y_i - w x_i)^2 / \partial w = 2 w x_i^2 - 2 x_i y_i$$

$$\frac{\partial g(w)}{\partial w} = 2 \sum_{i \in [m]} w x_i^2 - x_i y_i$$

The second derivative is positive, so extrema will be min.

$$\frac{\partial g(w)}{\partial w} = 0 \Leftrightarrow w = \frac{\sum_{i \in [m]} x_i y_i}{\sum_{i \in [m]} x_i^2}$$

This can be generalized to higher dimensions.

Next time: what if we don't have realizability? Convex loss functions, stochastic gradient descent.

4.6 Section

$$HS_d = \{\text{sign}(H_{wb}(x))\}$$

Alg. Input: training set S. $w^0 = 0$ while true: for $i \in [m]$ if $y_i\langle w^t, x_i\rangle \leq 0$ then $w^{t+1} \leftarrow w^t + y_i x_i$ and $t \leftarrow t+1$. If no update in 1 interation of i then break.

Proof. Update will occur in direction of error. $\langle w, x\rangle < 0$ therefore $y = \text{sign}(\langle w^*, x\rangle) = 1$. $w \leftarrow w + x^\top y = w + x$. $R = \max_{i\in[m]} |x_i|$, $B = \min_w\{|w| : \langle w, x\rangle y \geq 1, \forall i\}$ satisfied by w^*.

Claim: Perceptron terminates in $\leq (RB)^2$ steps.

Big Idea: $\cos(w^t, w^*) \leq 1$, $= \langle w^t, w^*\rangle/(||w^t||||w^*||)$.

Induct on see iPhone photos.

$||w^{t+1}||^2 = ||w^t + y_i x_i||^2 = ||w^t||^2 + \langle w^t, y_i x_i\rangle + ||y_i \cdot x_i||^2 = ||w^t||^2 + 0 + R^2$. $||w^0|| = 0 \Rightarrow ||w^t|| \leq tR^2$.

$$1 \leq \frac{\langle w^t, w^*\rangle}{||w^t||||w^*||} \leq t/\sqrt{t}RB$$

$$\Rightarrow \sqrt{t}/RB \leq 1 \Rightarrow t \leq (RB)^2$$

Mistake bound. Perceptron is a mistake bound. □

Polynomial expansion: $x_i = (x_{i1}, x_{i2})$ to $x_i' = (x_{i1}^2, x_{i2}^2, x_{i1}x_{i2}, x_{i1}, x_{i2}, 1)$.

Exercise 1 in 9.6.

$$L_S = \sum_{i\in m} |w \cdot x_i - y_i|$$

$$s_i = |w \cdot x_i - y_i|$$

$$s_i \geq w \cdot x_i - y_i$$

$$-s_i \leq w \cdot x_i - y_i$$

$$x = (w_1, \ldots, w_d, s_1, \ldots, s_m)$$

See iPhone photos

5 Convex learning problems

5.1 Recap

Linear predictors have form $\phi \circ f(x)$ where $f(x) = \mathbf{w}^\top + b$ $\phi : \mathbb{R} \to \mathbb{R}$. Examples:

- Halfspaces, where $\phi = \text{sign}$
- Linear regression $f(x) = w^\top x$.

Theorem: VC dimension of halfspaces is d therefore they are efficiently learnability (small sample complexity). Furthermore, in the separable case half spaces are computationally efficiently learnable. Computational feasabiliy: perceptron, LP can solve

$$ERM_H(S) = \operatorname*{argmin}_{w \in \mathbb{R}^d} \sum_{i \in [m]} |\text{sign}(w^\top x) - y_i|$$

in polynomial time.

Linear regression: using least squared error. Doing ERM on this hypothesis class can be done analytically.

5.2 Convex loss functions

Convex loss functions are desireable.

5.2.1 Logistic regression

Sometimes data is not separable, so linear classifiers will be computationally hard to learn.

We learn hypothesis $h : \mathbb{R} \to [0, 1]$. $h(x)$ is the probability that x is 1. ϕ is sigmoid.

$$\phi(z) = \frac{1}{1 - e^{-z}}$$

So we let

$$h_w(x) = \frac{1}{1 + e^{-w^\top x}}$$

Properties: $h_w(x)$ should be large when $y = 1$ and $1 - h(x)$ should be large when $y = -1$. Rearranging $1 - h_w(x)$ we get

$$1 - h_w(x) = \frac{1}{1 + e^{w^\top x}}$$

and we want a loss fn that will monotonically increase with

$$\frac{1}{1 + \exp(y(w^\top x))}$$

and thus should increase monotonically with $1 + \exp(-y(w^\top x))$. Let us choose logistic loss

$$\log(1 + \exp(-y(w^\top x)))$$

Therefore ERM for logisitc regression is

$$\operatorname{argmin} \frac{1}{m} \sum_{i \in [m]} \log(1 + \exp(-y_i(w^\top x_i)))$$

This loss is nice because it is convex and therefore ideal for stochastic gradient descent.

5.2.2 Elements of convex loss functions

Definition 9 (Convex set)**.** A set C is called convex if for any $x_1, x_2 \in C$ we have that $\lambda x_1 + (1-\lambda)x_2 \in C$ for $\lambda \in [0,1]$, or in other words, the line between any two points in the set is in itself the set.

Definition 10 (Convex function)**.** for a convex set C we say that $f : C \to \mathbb{R}$ is **convex** if for any $x_1, x_2 \in C$, $\lambda \in [0,1]$:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$

Nice properties: A local minimum of a convex function is also the global minimum.

Definition 11 (Local optimum)**.** For a minimization problem, a solution $x^* \in C$ is locally optimal if there exists $\epsilon > 0$, $f(x^*) \leq f(x)$ for all $x \in C$ s.t. $||x - x^*||_2 < \epsilon$.

Theorem 7 (Local optimums of convex functions are global optimums)**.** Let C be convex set, $f : C \to \mathbb{R}$ is convex then if x^* is locally optimal, it is also globally optimal.

Proof. For a contradiction, assume x^* is locally optimal but there is some $y \in C$ s.t. $f(y) < f(x^*)$. Let $z = \lambda x^* + (1-\lambda)y$.

$$\begin{aligned} f(z) &= f(\lambda x^* + (1-\lambda)y) \\ &\leq \lambda f(x^*) + (1-\lambda)f(y) \\ &< \lambda f(x^*) + (1-\lambda)f(x^*) \\ &= f(x^*) \end{aligned}$$

Therefore $f(z) < f(x^*)$ which contradicts local optimality of x^*. □

Properties of convex functions:

- For every u, $f(u) \geq f(w) + \langle \nabla f(w), u - w \rangle$. (all points can be bounded below by a linear fn)
- $\nabla f(w) = \begin{bmatrix} \frac{\partial f(w)}{\partial w_1} \\ \dots \\ \frac{\partial f(w)}{\partial w_d} \end{bmatrix}$

Lemma 7.1. Let $f : \mathbb{R} \to \mathbb{R}$ be a twice differentiable fn. Then the following are equivalent.

1. f is convex
2. f' is monotonically nondecreasing
3. $f'' \geq 0$

Example 6.

- $f(x) = x^2$ is convex. Therefore $f'(x) = 2x$ which is monotonically nondecreasing. $f''(x) = 2 \geq 0$.
- $f(x) = \log(1 + \exp(x))$ is convex. $f'(x) = 1/(\exp(-x) + 1)$ is monotonically nondecreasing.

Claim 1. Assume that $f : \mathbb{R}^d \to \mathbb{R}$ is $f(w) = g(w^\top x + y)$ for some $x \in \mathbb{R}^d$, $y \in mbbR$, and g is convex, then f is also convex.

Proof.

$$\begin{aligned} f(\lambda w_1 + (1-\lambda)w_2) &= g(\langle \lambda w_1 + (1-\lambda)w_2, x \rangle + y) \\ &\leq \lambda g(w_1^\top x + y) + (1-\lambda) g(w_2^\top x + y) \end{aligned}$$

□

Example 7 (Square and logistic loss functions are convex)**.** $f(w) = (w^\top x - y)^2$ is convex (squared loss). $f(w) = \log(1 + \exp(-y\langle w, x \rangle))$ is convex when $y \in \{-1, 1\}$ (logistic loss).

Claim 2. Let f_1, f_2 be convex functions $f_i : \mathbb{R}^d \to \mathbb{R}$ then the following funcitons are alos convex:

- $g(x) = max_{i \in [r]} f_i(x)$
- $g(x) = \sum_{i \in [r]} \alpha_i f_i(x)$ where $\alpha \geq 0$ for all $i \in [r]$.

Proof. See p. 160 of UML □

Example 8 (Absolute value function)**.** $g(x) = |x|$. $g(x) = \max(-x, x)$. x and $-x$ are convex, thus g is convex.

Definition 12 (ρ-Lipschitz)**.** Let $C \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is ρ-Lipschitz over C if for all $w_1, w_2 \in C$

$$||f(w_1) - f(w_2)|| \leq \rho ||w_1 - w_2||$$

Example 9. $f(x) = |x|$ is 1-Lipschitz over $\mathbb{R}$ because $|x_1 - x_2| \leq |x_1 - x_2|$.

$f(x) = \log(1 + \exp(x))$ is 1-L: $|f'(x)| = |1/(\exp(-x) + 1) \leq 1$.

$f(x) = x^2$ is not ρ-L over $\mathbb{R}$ for any ρ, but it is ρ-L over $C = \{x : |x| \leq \rho/2\}$.

$f(x) = v^\top x + b$ is $||v||$-L over $\mathbb{R}$.

Claim 3. Let $f(x) = g(h(x))$ where g is a-L and h is b-L, then $f(x)$ is ab-L.

Proof. Immediate from definition. □

Example 10. h is a linear function, then $f(x)$ is $(a||v||)$-L.

Definition 13 (β-smooth)**.** A differentiable fn $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth if $||\nabla f(x_1) - \nabla f(x_2)|| \leq \beta ||x_1 - x_2||$.

Claim 4. If f is β-smooth then for all $v, w \in C$,

$$f(v) \leq f(w) + \langle \nabla f(w), v - w \rangle + \frac{\beta}{2}||v - w||^2$$

Also, $||\nabla f(w)||^2 \leq 2\beta f(w)$.

Claim 5. Let $f(w) = g(w^\top x + b)$ where g is β-smooth then f is $\beta||x||^2$-smooth.

Next week: gradient descent.

5.3 Section

$H \in \mathbb{R}$, $l(w, (x, y)) = (wx - y)^2$ is not PAC learnable. See 12.8 in UML.

6 Gradient descent

6.1 Recap

Linear regression

Definition 14 (Convex learning problem)**.** A learning problem is convex if the hypohesis class H is a convex set and for all $z \in Z$, $l(\cdot, x)$ is a convex function.

Lemma 7.2. For a convex learning problem H, Z, l ERM reduces to a convex optimization problem.

Proof. Given $S = \{(x_i, y_i)\}_{i=1}^m = \{z_i\}_{i=1}^m$, for all $h \in H$

$$L_S(h) = \frac{1}{m} \sum_{i \in [m]} l(h, z)$$

is a convex function.

$$ERM_H(S) = \operatorname*{argmin}_{h \in H} L_S(h)$$

□

6.2 Gradient descent

Initialize: $w^1 \leftarrow 0$, η, T. For $t = 1, \ldots, T$ do: compute $\nabla f(w^t)$, $w^{t+1} \leftarrow w^t - \eta \nabla f(w^t)$. Return $\bar{w} = \frac{1}{T} \sum_{i \in [T]} w^t$.

We return the average?

Theorem 8 (Gradient descent on ρ-Lip loss functions)**.** Let f be a convex function that is ρ-Lipschitz, then running gradient descent with step size $\eta = \sqrt{B^2/(\rho^2 T)}$ and $T \geq B^2\rho^2/\epsilon^2$ then $f(\bar{w}) - f(w^*) \leq \epsilon$ where $w^* \in \operatorname{argmin}_{w:||w|| \leq B} f(w)$.

Proof.

$$\begin{aligned}
f(\bar{w}) - f(w^*) &= f((1/T) \sum_{t \in [T]} w^t - f(w^*) \\
&\leq (1/T) \sum f(w^t) - f(w^*) \\
&= (1/T) \sum (f(w^t) - f(w^*)) \\
&\leq (1/T) \sum \langle w^t - w^*, \nabla f(w^t) \rangle \\
&\leq B\rho/\sqrt{T} \qquad \text{by the following lemma}
\end{aligned}$$

□

Lemma 8.1. Let $v_1, \ldots, v_T$ be an arbitrary sequence of vectors, $w^1 = 0$,

and update rule is $w^{t+1} = w^t - \eta v_t$ then

$$\sum_{t\in[T]} \langle w^t - w^*, v_t\rangle \leq \frac{||w^*||^2}{2\eta} + (\eta/2)\sum ||v_t||^2$$

for $||v_t|| \leq \rho$ (ρ-Lipschitz), $\eta = \sqrt{B^2/\rho^2 T}$, $||w^*|| \leq B$:

$$1/T \sum \langle w^t - w^*, v_t\rangle \leq B\rho/\sqrt{T}$$

Proof.

$$\begin{aligned}\langle w^t - w^*, v_t\rangle &= (1/\eta)\langle w^t - w^*, \eta v_t\rangle \\ &= (1/2\eta)\left(-||w^t - w^* - \eta v_t||^2 + ||w^t - w^*||^2 + \eta^2||v_t||^2\right) \\ &= (1/2\eta)(-||w^{t+1} - w^*||^2 + ||w^t - w^*||^2) + (\eta/2)||v_t||^2\end{aligned}$$

...

$$\begin{aligned}\sum \langle w^t - w^*, v_t\rangle &\leq \frac{||w^*||^2}{2\eta} + (\eta/2)\sum ||v_t||^2 \\ &\leq \frac{||w^*||^2}{2\eta} + (\eta/2)T\rho \\ &\leq \dots \\ &\leq B\rho/\sqrt{T}\end{aligned}$$

(Telescoping sum) Choosing $T \geq B^2\rho^2/\epsilon^2$ we get our proof. □

6.3 Stochastic gradient descent

Computing $\nabla f(w^t)$ is hard to compute with *all* your data. Very costly.

Initialize: $w^1 \leftarrow 0$, η, T. For $t = 1, \dots, T$ do: compute $\widetilde{\nabla} f(w^t)$, $w^{t+1} \leftarrow w^t - \eta \nabla f(w^t)$. Return $\bar{w} = \frac{1}{T}\sum_{i\in[T]} w^t$.

Definition 15 (Unbiased estimator)**.** $\widetilde{\nabla} f(w)$ is an unbiased estimator of the gradient of f of w $\nabla f(w)$ if $\mathbb{E}[\widetilde{\nabla} f(w)] = \nabla f(w)$.

For our loss, $\nabla f(w) = (1/m) \sum_{i \in m} \nabla l(w, (x, y))$. If m is large, this is a big sum. Why don't we just sample uniformly at random and use a subset of our sample with size $n < m$. We will still converge to the minimum, but drunkenly (not in the most efficient path.) Rate of convergence is slower, but is cheap and still works.

Theorem 9 (Stochastic gradient descent on ρ-Lipschitz loss functions)**.** Let f be a convex function that is ρ-Lipschitz, then running stochastic gradient descent with step size $\eta = \sqrt{B^2/(\rho^2 T)}$ and $T \geq B^2 \rho^2 / \epsilon^2$ then $\mathbb{E}[f(\bar{w})] - f(w^*) \leq \epsilon$ where $w^* \in \text{argmin}_{w:||w|| \leq B} f(w)$.

$$ERM_H(S) \in \underset{h \in H}{\text{argmin}}\, L_S(h) \approx \underset{h \in H}{\text{argmin}}\, L_D(h)$$

This is big. It means optimization is approximately machine learning.

7 Surrogate loss functions, SVMs, regularization

7.1 Surrogate loss functions

What happens if l is not convex? Implementing ERM can be computationally hard.

Example 11 (0-1 loss)**.**

$$l^{0-1}(w, (x, y)) = 1y \neq \text{sign}(\langle w, x \rangle)$$

Approach: use a surrogate loss function that

1. Is convex

2. Has an upper bound on the original loss.

Hinge loss:

$$l^{hinge}(w,(x,y)) = \max(0, 1 - y\langle w, x\rangle)$$

Generalization:

$$L_{\mathcal{D}}^{hinge}(A(S)) \leq \min_{w\in\mathcal{H}} L_{\mathcal{D}}^{hinge}(w) + \epsilon$$

Since surrogate:

$$L_{\mathcal{D}}^{0-1}(A(S)) \leq L_{\mathcal{D}}^{hinge}(A(S))$$

Therefore

$$L_{\mathcal{D}}^{0-1}(A(S)) \leq \min_{w\in\mathcal{H}} L_{\mathcal{D}}^{hinge}(w) + \epsilon$$

Thus

$$L_{\mathcal{D}}^{hinge}(A(S)) \leq \min_{w\in\mathcal{H}} L_{\mathcal{D}}^{0-1}(w) + (\min_{w\in\mathcal{H}} L_{\mathcal{D}}^{hinge}(w) - \min_{w\in\mathcal{H}} L_{\mathcal{D}}^{0-1}(w)) + \epsilon$$

First term: Approximation error. Second term: Optimization error. Third term: Estimation error.

7.2 SVMs

Support vector machines.

Example 12. (Assuming separable case) there are multiple ERM solutions. We want to select the one such that the distance to each example is maximized.

Main idea: the true error of a halfspace can be bounded in terms of the margin over the training sample.

Definition 16 (Lineary separable)**.** A training set S consisting of m points is linearly separable if there exists some halfspace (w, b) such that $y_i = \text{sign}(\langle w, x_i\rangle + b)$.

Claim 6. The distance between a point $x \in \mathbb{R}^d$ and hyperplane (w, b) with $\|w\| = 1$ is $|\langle w, x\rangle + b|$.

7.2.1 Hard-SVM

$$\max_{(w,b):\|w\|=1} \min_{i\in[m]} |\langle w, x_i\rangle + b|$$

such that for all i, $y_i(\langle w, x_i\rangle + b) \geq 0$. Simplifying we get the equivalent problem

$$\max_{(w,b):\|w\|=1} \min_{i\in[m]} y_i(\langle w, x_i\rangle + b)$$

Finally, we have the equivalent

$$\operatorname*{argmin}_{(w,b)} \|w\|^2$$

such that $y_i(\langle w, x\rangle + b) \geq 1$. This is a linear programming problem.

7.2.2 Hard SVM sample complexity

Recall that the VC dimension of halfspaces is $d + 1$. Also, the fundamental theorem of learning: if m much smaller than d/ϵ cannot learn ϵ-accurate halfspaces. So what if d is very large?

Definition 17 (Margin)**.** A distribution $\mathcal{D}$ over examples $\mathbb{R}^d \times \{-1, 1\}$ is separable with a (γ, ρ)-margin if there exists (w^*, b^*) such that $\|w\| = 1$ and $y(\langle w^*, x\rangle + b^*) \geq \gamma$ and $\|x\| \geq \rho$ with probability 1 over $(x, y) \sim \mathcal{D}$.

Theorem 10. Let $\mathcal{D}$ be a distribution that is separable with a (γ, ρ)-margin. Then, with probability $1 - \delta$ over training sample S of size m,

$$L^{0-1}_{\mathcal{D}}(\text{Hard-SVM}(S)) \leq \sqrt{\frac{4(\rho/\gamma)^2}{m}} + \sqrt{\frac{2\log(2/\delta)}{m}}$$

This is surprising because the it does not depend on d.

What if the data is not separable? Soft-SVMs coming soon.

7.3 Regularization

We have training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ with samples in $\mathbb{R}^d \times \{-1, 1\}$.

$$RLM = \underset{w}{\operatorname{argmin}}(L_S(w) + R(w))$$

Where R is the complexity of your learned w to prevent over-fitting.

Tikhonov regularization:

$$R(w) = \lambda \|w\|^2$$

$$RLM(S) = \underset{w}{\operatorname{argmin}}(L_S(w) + \lambda \|w\|^2$$

This can be thought of as maximizing stability. Stability: (informally) a learning algorithm is stable if changing the input a little does not affect the output much.

Overfitting: $\mathbb{E}_S[L_{\mathcal{D}}(A(S)) - L_S(A(S))]$.

Notation: Given $S = (z_1, \ldots, z_m)$ and z', let

$$S^{(i)} = (z_1, \ldots, z_{i-1}, z', z_{i+1}, \ldots, z_m)$$

Theorem 11. Let $S = (z_1, \ldots, z_m)$ be an iid sequence of examples from $\mathcal{D}$ and let $z' \sim \mathcal{D}$. Then for any learning algorithm A,

$$\mathbb{E}_{S \in \mathcal{D}}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] = \mathbb{E}_{(S,z') \sim \mathcal{D}^{m+1}}[l(A(S^{(i)}, z_i)) - l(A(S), z_i)]$$

where $i \sim U(m)$.

Proof.

$$\begin{aligned}\mathbb{E}_S[L_D(A(S))] &= E_S[E_{z'\sim\mathcal{D}}[l(A(S), z')]] \\ &= E_{S,z'\sim Dm+1}[l(A(S), z')] \\ &= E_{S,z',\sim D\times U(m)}[l(A(S^{(i)}), z_i)]\end{aligned}$$

$$\begin{aligned}\mathbb{E}_S[L_S(A(S))] &= \mathbb{E}_S[1/m \sum_{i\in[m]} l(A(S), z_i)] \\ &= \mathbb{E}_{S,i}[l(A(S), z_i)]\end{aligned}$$

□

Definition 18. A learning algorithm A is on-average-replace-one-stable with rate $\epsilon(m)$ if for all distributions $\mathcal{D}$ stability is at most $\epsilon(m)$

$$\mathbb{E}[\ldots] \leq \epsilon(m)$$

See previous section for contents of $\mathbb{E}[\ldots]$.

8 Apocalypse

No notes due to COVID-19 moveout.

9 Neural Networks

9.1 Recap

Logistic regression:

$$\Phi_{sig}(Z) = 1/(1 + \exp(-Z))$$

$$H_{sig} = \{x \to \Phi_{sig}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathbb{R}^d\}$$

Maximum likelihood objective ($y \in \{0, 1\}$)

$$\max \Pr((\mathbf{x}, y)|\mathbf{w}) = \Phi_{sig}(x)^y (1 - \Phi_{sig}(x))^{1-y}$$

$$= \min \log(1 + \exp(-y' \langle \mathbf{w}, \mathbf{x} \rangle))$$

$y \in \{-1, 1\}$ Logistic loss or cross-entropy loss.

9.2 Logistic loss to neural networks

Rewriting logistic regression for neural nets.

$$\hat{y} = \sigma(\mathbf{x}, \mathbf{w})$$

1-layer neural net. For a 2-layer,

$$\hat{y} = \sigma(\sigma(\mathbf{x}, \mathbf{w}^{(0)}), \mathbf{w}^{(1)})$$

$\mathbf{w}^{(0)} \in \mathbb{R}^{d \times h}$, $\mathbf{w}^{(1)} \in \mathbb{R}^h$ σ can represent other functions than the cross-entropy loss. Examples include ReLU $\max(0, x)$, tanh, 0-1, and more.

Another interpretation of neural nets is a logistic regression on a mapping from the input to some hidden dimension. The goal is to separate the data in the hidden dimension linearly.

9.3 Expressivity of NNs: how powerful is a neural net?

Claim 7. A neural net of depth 2 can express any boolean function.

Example 13. Let $d = 4$.

x_1	x_2	x_3	x_4	y
0	0	1	1	1
...				

Writing this in disjunctive normal form (or of ands)

$$y = \bar{x}_1 \bar{x}_2 x_3 x_4 + \dots$$

We can represent any function with 2^{d-1} hidden nodes.

This good because we can represent any function. This is bad because it requires exponential nodes. This can be solved with deep neural networks (hidden layers at least 3.)

Depth 1 cannot express XOR. Depth 2 can express XOR with 3 nodes. A neural network can represent the XOR of d vars in $3(3-d)$ nodes using in $2 \log d$ layers.

Theorem 12 (Claude Shannon)**.** There exist functions that require 2^{d-1} nodes regardless of depth.

Theorem 13 (Universal approximation theorem (1989))**.** A NN of depth 2 can approximate any continuous function.

Proof. For $d = 1$, x, thresholds T_1 and T_2, $T_1 > T_2$ gives an impulse of length $T_1 - T_2$. Divide function into chunks and design pulses for each chunk. Thus a 1-layer network with infinite nodes can mimic a function.

Similar methods for higher dimensions. □

9.4 Sample and time complexity

As shown in UML the sample complexity is

$$O(|E| \log |E|)$$

Where E is the set of parameters. This assumes small number of parameters. This less relevant because of overparameterization, where you have more parameters than data points. Run time: ERM for sign-gated NNs is NP-hard. How do we train then? Treat it like a convex problem and do SGD.

9.5 Forward propagation and backpropagation

Recall gradient descent: init $\mathbf{x}^0$ randomly. $k = 0$ (count). While $f(x^k) - f(x^k - 1) > \epsilon$ or $\|\nabla_x f(x^k)\| > \epsilon$, do: $k = k + 1$, $x^{k+1} = x^k - \eta \nabla_x f(x^k)$.

For neural nets, loss $L(y, \hat{y}) = \frac{1}{m} \sum l(y_i, \hat{y}_i)$. Init w^k for layers $k = 1, \dots K$. While loss not converge, do for $k = 1, \dots, K$,

$$w_k = w_k - \eta \nabla_{w_k} L(y, \hat{y}) = w_k - \frac{\eta}{m} \sum_i \nabla_{w_k} l(y_i, \hat{y}_i)$$

Forward step: feed the input.

$$y_i^{(0)} = x_1$$

for $k = 1 \dots N$, for $j = 1, \dots, W$ where W is the kth layer width.

$$z_j^{(k)} = \sum_i w_{ij}^{(k)} y_i^{(k-1)}$$

$$y_h^{(k)} = f_k(z_j^{(k)})$$

Backward pass: taking the gradient. Training and updating weights.

$$y^{(N)} : \frac{\partial l}{\partial \hat{y}_i} = \frac{\partial l}{\partial y_i^{(N)}}$$

$$z^{(N)} : \frac{\partial l}{\partial z_i^{(N)}} = \frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} \frac{\partial l}{\partial y_i^{(N)}}$$

$$\frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} = f'_N(z_i^{(N)})$$

$$w^{(N)} : \frac{\partial l}{\partial w_{11}^{(N)}} = \frac{\partial z_i^{(N)}}{\partial w_{11}^{(N)}} \frac{\partial l}{\partial z_1^{(N)}}$$

$$= y_1^{(N-1)}$$

See the book good grief.

10 k-clustering

Give a set of points X in some large space X'.

Goal: partition points in X into clusters $c1, \ldots, c_k$ in a way that minimizes a clustering objective.

10.1 Finding a good objective is non-trivial

- A good objective is not universal.

  Example 14. Say that $k = 2$. Objective is that all points close to each other are in the same cluster. Another objective is that the furthest points in a cluster are not far from each other.

- There is no ground truth.

10.2 k-means objective

In k-means objective we have centroids.

$$\mu(C_i) = \operatorname*{argmin}_{\mu \in \mathcal{X}'} \sum_{k \in C_i} d(x, \mu)^2 = \frac{1}{C_i} \sum_{x \in C_i} x$$

where $\mathcal{X} \subseteq \mathcal{X}'$ is a metric space with distance function d.

Definition 19 (k-means objective)**.** Given clusters $C_1, \ldots, C_k$, the k-means objective is

$$F(C_1, \ldots, C_k) = \sum_{i \in [k]} \sum_{x \in C_i} \|x - \mu(C_i)\|^2$$

This problem is actually an NP-hard optimization problem. There exists, however, a popular heuristic algorithm that works pretty well in practice.

Input: points $X \subseteq \mathbb{R}^n$, number of clusters k

Initialization: Select random centroids $\mu_1, \ldots, \mu_k \in \mathcal{X}'$

Repeat until convergence: $\forall i \in [k]$ set $C_i = \{x \in \mathcal{X} : i = \text{argmin}\|x - \mu_j\|\}$ (not sure about notation here, but the idea is we put the closest xs to μ_i in the cluster. $\forall i \in [k]$ update $\mu_i = \frac{1}{|C_i|} \sum x \in C_i x$.

Return $C_1, \ldots, C_k$.

Theorem 14. In the k-means algorithm, at each step the solution is *weakly improved*, i.e. in every step t

$$F(C_1^{(t)}, \ldots, C_k^{(t)}) \leq F(C_1^{(t-1)}, \ldots, C_k^{(t-1)})$$

where $C_i^{(t)}$ is the cluster C_i at iteration $t \leq T$.

Proof. Let μ_i^t be the centroid of C_i at iteration t. Take $F(C_1^t, \ldots)$ is less than the sum with $\mu_i^t \leftarrow \mu_i^{t-1}$ and then replace C_i^t with C_i^{t-1}, yielding larger quantities both times. This last replacement yields $F(C_1^{t-1}, \ldots)$. □

10.3 Other clustering objectives

- k-means: as we have seen.
- k-medioid: restricting mu_i to a point in the dataset.
- k-median: same as mediod without squaring distances.

11 Unsupervised learning

Here are some examples of unsupervized learning problems. Document summary: given a document, select k sentances that summarize the document. Feature selection: given a lots of data, which k types of data will make prediciton possible? Image compression: given a grid of pixels make blocks and choose k pixels that represent the block. Ranking: given webpages with keywords and a query, choose k webpages that represent all the pages returned by the query.

These can be solved using submodular optimization.

12 Max cover

Discs (n of them) covering points.

Goal: Select k discs that together cover as many points as possible. In other words find

$$S \in \operatorname*{argmax}_{R:|R|\leq k} |\bigcup_{i\in R} a_i|$$

where $a_1, \ldots, a_n$ is the set of discs.

Notation: $f(s)$ in the number of points covered by S. $f_S(R)$ is the number of points covered by R not covered by S, also known as the marginal contribution of R.

Greedy algorithm: Select at each step the disc with largest marginal contribution and add it to the set. ($S \leftarrow S \cup \text{argmax}_a f_S(a)$). Return S.

This algorithm (greedy) is not optimal. It is possible to construct a problem that will trick the algorithm. It turns out that max cover is NP-hard.

Theorem 15. For any instance of max cover, let S be the solution returned by the greedy algorithm and let O be the optimal solution. It holds that $f(S) \geq (1 - 1/e)f(O)$, or at least 63% correct.

Proof.

Fact 15.1 (f_S is subadditive)**.** In max-cover, for any $S \in \mathbb{N}$, f_S is *subadditive*, or in other words

$$f_S(X \cup Y) \leq f_S(X) + f_S(Y)$$

Notation: a_i is the disc selected at step i of Greedy. $S_i = \{a_1, \ldots, a_i\}$.

Lemma 15.1. At every step $i \in \{0, 1, \ldots, k\}$ of Greedy: $f(S_{i+1}) - f(S_i) \geq \frac{1}{k}(f(O) - f(S_i))$. The left side is the marginal contribution of adding disc a_{i+1}. The right side is $\frac{1}{k}$th of the difference between the optimal and current solution.

Proof. $O = \{o_1, o_2, \ldots, o_k\}$. $o^* \in \text{argmax}_{o \in O} f_{S_i}(o)$.

$$f_{S_i}(O) \leq \sum_{o \in O} f_{S_i}(o) \leq k f_{S_i}(o^*) \leq k(f(S_{i+1}) - f(S_i))$$

by the subadditive property. Therefore we have the lemma. □

Lemma 15.2. At every step $i \in \{1, \ldots, k\}$ we have

$$f(S_i) \geq (1 - (1 - \frac{1}{k})^i) f(O)$$

Proof. By induction on i, at $i = 1$ the proof follows by the previous lemma. $f(S_{i+1}) - f(S_i) \geq \frac{1}{(}(f(O) - f(S_i)))$ which gives us

$$f(S_i) \geq (1 - 1(1 - \frac{1}{k})) f(O)$$

now assume that for $i \in \{1, \ldots, l\}$ our lemma holds. Now we prove for $l + 1$.

$$\begin{aligned} f(S_{l+1}) &\geq \frac{1}{k}(f(O) - f(S_l)) + f(S_l) \\ &= \frac{1}{k} f(O) + (1 - \frac{1}{k}) f(S_l) \\ &\geq \frac{1}{k} f(O) + (1 - \frac{1}{k})(1 - (1 - \frac{1}{k})^l) f(O) \\ &= (1 - (1 - \frac{1}{k})^{l+1}) f(O) \end{aligned}$$

Thus by induction we have the lemma. □

Now the proof is done because for $k \geq 1$, $(1 - \frac{1}{k})^k \leq 1/e$. Thus for $i = k$ by the lemmas $f(S_k) \geq (1 - \frac{1}{e}) f(O)$. □

13 Unsupervised learning as max cover

Unsupervised learning problems are often maximization under cardinality constraint. The sum is less than the total of its parts.

13.1 Submodularity and examples

Definition 20 (Submodularity)**.** Given a set of elements

$$N = \{a_1, \ldots, a_m\}$$

a function $f : 2^N \to \mathbb{R}$ is *submodular* if for all $S \subseteq T$ and $a \in N \setminus T$ we have

$$f(S \cup a) - f(S) \geq f(T \cup a) - f(T)$$

In other words the marginal cost of adding elements to the set decreases monotonically as the size of the set increases. Equivalently, a function f is submodular if for all S, T, $f(S \cup T) \leq f(S) + f(T) - f(S \cap T)$.

Some cannonical examples of submodular functions include

- Single peak: $f(S) = \max_{a \in S} f(a)$
- Additive functions: $f(S) = \sum_{a \in S} f(a)$
- Coverage functions: $f(S) = |\bigcup_{i \in S} a_i|$ when $a_1, \ldots, a_n$ are discs containing points in some universe.
- Diversity: $f(S) =$ number of colors in S.
- Cut function: $f(S)$ is the number of edges between S and $V \setminus S$ on $G = (V, E)$.

These problems are about increasing entropy in the set.

Definition 21 (Monotonicity)**.** A function $f : 2^N \to \mathbb{R}$ is *monotone* if $S \subseteq T$ then $F(S) \leq f(T)$.

There are submodular functions that are not monotone. For instance the cut function.

The problem we solved last time was essentially 'given $f : 2^N \to \mathbb{R}$ find k elements that maximize f. In other words find $S = \text{argmax}_{T:|T|\leq k} f(T)$.'

Theorem 16 (Greedy approximation of monotone submodular functions)**.** For any monotone submodular function $f : 2^N \to \mathbb{R}$ running the greedy algorithm for k steps returns a solution $S \leq 1$ such that $f(S) \leq (1 - \frac{1}{e}) \max_{T:|T|\leq k} f(T)$.

Proof. This follows almost word for word from the theorem of the previous section. □

13.2 Clustering as a submodular objective

Goal: find k representatives points in $\mathcal{X}$.

$d(x, u)$ is the distance between x and u. $S = \{u_1, \ldots, u_n\}$. $L(S) = \frac{1}{|X|} \sum_{x\in X} \min_{u\in S} d(x, u)$. Let $f(S) = -L(S)$. We now have a maximization problem. We add a point x_0 to the data set and set $f(S) = L(x_0) - L(S)$ to make f always positive. We are still not done because f is not normalized.

Definition 22 (Normalization)**.** A function f is *normalized* if $f(\emptyset) = 0$.

We finally have a monotone submodular normalized funtion f defined as

$$f(S) = L(x_0) - L(S \cup \{x_0\})$$

If the function is not monotone, we can get arbitrarily bad results, but randomizing can give us good results.